C S 26A: RUBY & FUNCTIONAL PROGRAMMING

Foothill College Course Outline of Record

Heading	Value
Units:	4.5
Hours:	4 lecture, 2 laboratory per week (72 total per quarter)
Advisory:	One of the following: C S 1A, 2A; not open to students with credit in C S 85A.
Degree & Credit Status:	Degree-Applicable Credit Course
Foothill GE:	Non-GE
Transferable:	CSU/UC
Grade Type:	Letter Grade (Request for Pass/No Pass)
Repeatability:	Not Repeatable

Student Learning Outcomes

- A successful student will be able to use functional paradigm to design and implement a clear, well-structured Ruby program. Specifically, the student will immutability, currying, recursion and lazy evaluation in his or her programs.

Description

Introduction to functional programming languages using Ruby as an educational and practical development environment. Students will learn how to create programs that use the functional paradigm while obeying the object-oriented structure inherent in the language. Many examples and topics will be covered including database-driven web applications using the Rails framework.

Course Objectives

The student will be able to:
A. Download and install a Ruby software development environment and extra tools for Rails development.
B. Produce clearly written code in an industry standard style appropriate for Ruby.
C. Write code using numeric, string and user-defined objects in Ruby, and explain variable scope.
D. Incorporate Ruby variable expressions and user interaction in a program to compute numeric and string results.
E. Define, analyze and code the basic Ruby constructs.
F. Design, test, and debug Ruby code blocks and methods to be used in programs, and demonstrate the way parameters are passed and results returned.
G. Write Ruby programs using object-oriented design, and contrast the difference between object-oriented and procedural code.
H. Construct collections of data using hashes and arrays.
I. Write Ruby programs using functional programming design, and discuss the differences and compatibility between functional programming and object-oriented design.
J. Use the exception mechanism to report errors.
K. Develop web applications in Ruby using the Rails framework.

Course Content

A. The Ruby Development Environment
1. Installing a Ruby Interpreter
2. The interactive Ruby environment (IRE)
3. Text editor choices for writing Ruby (.rb) source code
4. The Rails installers
B. Coding Standards, Conventions and Styles
1. Acceptable indentation options
2. Naming conventions for Ruby variables, methods and classes
3. Line (#) comments and block (=begin/=end) comments
C. Ruby Data Objects
1. Integer and floating-point literals and objects
2. String literals and objects
3. Variables and barewords
4. Environment variables
5. Global and local scope
D. Expressions and User-Interaction
1. Using gets and puts to interact with the user
2. Assignment statements
3. Numeric operators and precedence
4. Logical operators
5. String comparison
6. Ruby symbols
E. Ruby Constructs
1. Selection (if, else, elseif, unless, case)
2. Loops (while)
3. Difference between FALSE and NIL
4. Nesting constructs
F. Code Blocks and Methods
1. Single-line and multi-line block
2. Implicit arguments and the yield key-word
3. Block variables and Procs
4. Using Ruby methods (puts, gets, etc.)
5. Defining Ruby methods
6. Parameters, functional returns, default parameters
7. Splat(*) arguments and bang(!) methods
8. Variable scope
9. Implicit objects and �self�
G. Object-Oriented Programming Using Classes and Methods
1. Encapsulation of member data
2. Encapsulation of member methods
3. Instance members and privacy
4. Open classes
5. Blocks
H. Collections of data
1. Arrays of objects
2. Arrays inside classes
3. The Hash class
I. Functional Programming
1. Immutability
2. Side effects
3. Higher-order functions
4. Currying
5. Recursion
6. Lazy evaluation
7. How Ruby encourages and implements functional programming principles
J. Error Reporting
1. Exceptions as error-reports
2. Method returns as error-reports
K. Web Development with Ruby on Rails

1. Rails overview
2. Rails script generators
3. Rails server
4. Controllers and views
5. Resources
6. Errors: actions, templates and routes
7. Writing forms applications
8. Migrations
9. Articles
10. Links

Lab Content

A. Familiarization with the beginning-level online lab environment
1. Download a Ruby development environment.
2. Modify and customize the settings of a Ruby Interpreter.
3. Use a text editor to create a Ruby program file.
4. Gain experience with the steps needed to edit a simple program.
B. Writing and troubleshooting simple programs using the Interactive Ruby Environment (IRE) and source files
1. Demonstrate the design and test cycle of a simple program using an IRE command-line environment.
2. Demonstrate the design and test cycle of a simple program using the text editor and source files.
3. Distinguish between syntax errors and logic errors.
4. Develop strategies for dealing with each type of error.
5. Debug code to produce a working program.
C. Exploring different object types using Ruby
1. Gain experience in effectively using Ruby to create code with numeric types.
2. Gain experience in effectively using Ruby to create code with string types.
3. Use Ruby to assist in defining and using compound data types.
4. Solve syntax and logic problems that arise from typical incorrect formulation of data types.
D. Demonstrating user interaction (I/O) and expressions
1. Play the role of user and programmer, alternately, to establish a user-interaction plan for a program.
2. Evaluate and comment on other students' user-interaction plan.
3. Use expressions to manipulate user-input data in Ruby programs.
4. Change modes from source code design (editing mode) to end-user interaction (run mode) in order to perform Q/A on the program.
5. Fix poor interaction behavior by adjusting source code and rerunning program until a satisfactory result is achieved.
E. Building a program that demonstrates �intelligence� though a combination of control statements
1. Become familiar with Ruby selection, loop and nesting to imbue a program with correct logic behavior.
2. Use functional programming to make programs robust.
3. Use structured programming to make control structures maintainable.
4. Run the program multiple times to verify that its control statements produce the correct behavior or output under any scenario.
5. Fix incorrect logic behavior by adjusting control structures and rerunning program until a satisfactory result is achieved.
F. Incorporating code blocks and methods in programming projects
1. Gain experience in writing a Ruby method.
2. Use a previously written method in a client program.
3. Refine methods/functions by adding or changing their definitions and observe the result.
4. Repeat steps above with code blocks.
5. Deduce the impact of a code block's or method's design on the programs that invoke it.
G. Building a program around object-oriented techniques
1. Use previously written classes to instantiate objects in program.
2. Use Ruby to assist in the creation of a programmer-defined class.
3. Demonstrate the correct choice of class members and methods for each class used.
H. Exploring Arrays and Hashes
1. Understand the proper use of Arrays and Hashes.
2. Incorporate an Arrays and Hashes into a program to facilitate the solution of an assigned problem.
3. Investigate use of variable indices and loops to shorten and clarify the logic in programs.
4. Use debugging techniques to solve problems that arise during the testing of a program.
I. Building a program around functional programming principles
1. Create programs that make use of the principle functional programming principles studied in the course.
2. Use Ruby to assist in the creation of functional programming components.
3. Demonstrate the correct choice of class members and methods for each class used.
4. Explain how functional programming techniques created more robust programs.
J. Installing and developing web applications in Ruby using Rails
1. Download and install a Rails development environment.
2. Configure the environment by starting the web server and using simple generators.
3. Make use of controllers and views in the Rails environment.
4. Interpret and fix errors in the project.

Special Facilities and/or Equipment

A. Access to a computer laboratory with Ruby and Rails development environments.
B. A website or course management system with an assignment posting component (through which all lab assignments are to be submitted) and a forum component (where students can discuss course material and receive help from the instructor). This applies to all sections, including on-campus (i.e., face-to-face) offerings.
C. When taught via Foothill Global Access on the Internet, the college will provide a fully functional and maintained course management system through which the instructor and students can interact.
D. When taught via Foothill Global Access on the Internet, students must have currently existing email accounts and ongoing access to computers with internet capabilities.

Method(s) of Evaluation

A. Tests and quizzes
B. Written laboratory assignments which include source code, sample runs and documentation
C. Final examination

Method(s) of Instruction

A. Lectures which include motivation for syntax and use of the Ruby language and functional programming concepts, example programs, and analysis of these programs.
B. Online labs (for all sections, including those meeting face-to-face/on campus) consisting of:
1. A programming assignment webpage located on a college-hosted course management system or other department-approved Internet environment. Here, the students will review the specification of each programming assignment and submit their completed lab work.

2. A discussion webpage located on a college hosted course management system or other department-approved Internet environment. Here, students can request assistance from the instructor and interact publicly with other class members.
C. Detailed review of programming assignments which includes model solutions and specific comments on the student submissions.
D. In-person or online discussion which engages students and instructor in an ongoing dialog pertaining to all aspects of designing, implementing and analyzing programs.
E. When course is taught fully online:
1. Instructor-authored lecture materials, handouts, syllabus, assignments, tests, and other relevant course material will be delivered through a college hosted course management system or other department-approved Internet environment.
2. Additional instructional guidelines for this course are listed in the attached addendum of CS department online practices.

Representative Text(s) and Other Materials

Michael Hartl. Ruby on Rails Tutorial: Learn Web Development with Rails. 2nd ed. Addison-Wesley, 2011. https://www.railstutorial.org/book
Although this text is older than the suggested "5 years or newer" standard, it remains a seminal text in this area of study.

Types and/or Examples of Required Reading, Writing, and Outside of Class Assignments

A. Reading
1. Textbook assigned reading averaging 30 pages per week.
2. Reading the supplied handouts and modules averaging 10 pages per week.
3. Reading on-line resources as directed by instructor though links pertinent to programming.
4. Reading library and reference material directed by instructor through course handouts.
B. Writing
1. Writing technical prose documentation that supports and describes the programs that are submitted for grades.

Discipline(s)

Computer Science